An Oriented Convergent Mutation Operator for Solving a Scalable Convergent Demand Responsive Transport Problem

Rémy Chevrier, Philippe Canalda, Pascal Chatonnay, Didier Josselin

An Oriented Convergent Mutation Operator for Solving a Scalable Convergent Demand Responsive Transport Problem

Rémy Chevrier[1], **Philippe Canalda**[2], **Pascal Chatonnay**[2] **and Didier Josselin**[1]

[1]Université d'Avignon et des Pays du Vaucluse UMR ESPACE (CNRS 6012)
74 rue Louis Pasteur, 84000 Avignon, France
[2]Laboratoire d'Informatique de Franche-Comté, Numerica
Cours Louis Leprince-Ringuet, 25201 Montbéliard, France
e-mail:[1]{firstname.lastname}@univ-avignon.fr -[2]{firstname.lastname}@univ-fcomte.fr

ABSTRACT

This paper presents a method for solving the convergence demand responsive transport problem, by using a stochastic approach based on a steady state genetic algorithm for enumerating a set of optimizing sprawling spanning trees, which constitute the best solutions to this problem. Specifically designed to speed up the convergence to optimal solutions, we introduce an oriented convergent mutation operator, allowing multi-objective considerations. So this solution lays the first stakes for considering real-time solving of such a problem. Led by computer science and geography laboratories, this study is provided with a set of experimental results evaluating the approach.
Keywords: Demand-Responsive Transport, Genetic Algorithm, Convergent Mutation Operator, Scalability, Real-Time

1. Introduction

The research deals with some arriving transportation services to be deployed combining regular transportation lines with Demand Responsive Transport (DRT). Indeed, the individual mobility demand leads to transport systems more and more reactive and efficient, tending to real-time processing. The transport supply must also respond to accurate customers.
Therefore, it becomes nowadays of primer interest to address several dimensions of the DRT economical and social efficiency. This may concern for instance the quality of service (QoS), the technological requirements due to pervasive systems, spatial partitioning for routing, flows management, vehicles and drivers allocation, optimal path finding, etc. We claim that these dimensions should be tackled in a global way, a non-efficiency of one of them restricting a global efficiency.
For instance, various relevant criteria can be raised. Some are related to geographical organization: territorial areas to be served, population density, number of mobility requests, spatial distribution of the clients, penetration rate of the service, target population. Some depend on the operating system itself: the stop points location, the access time to the service, its functioning (regular, demand-responsive, enabling pick-up and delivery, convergent or not...), vehicle types and capacities, line frequencies. Others allow to calculate specific clues, such as economical ratios.
If we focus on the subsequent objectives, we could define a list of combined tasks : minimizing the distances, costs and the number of vehicles, optimizing their capacity and their location on the served territory, optimizing the line frequencies and driver allocation, maximizing the QoS (minimizing the lost times, respecting the service agreements, comfort).
The experiment of our innovating transportation supply involves Transport Authorities (Communauté d'Agglomération du Pays de Montbéliard) and private carriers (Compagnie des Transports de Montbéliard), and also researchers from several laboratories in Geography and Computer Science, joined together in the TADvance pluridisciplinary group. The DRT concepts are developed within such an operational context. The partners share a common purpose whose implementation and tests are foreseen next year.
More precisely, we propose a solution for the Convergent DRT Problem (CDRTP) which is linked to different alternatives of the Pick-Up & Delivery N-Traveling Salesman Problem (P&D N-TSP) and integrable to assisted booking or operating systems. Actually, the CDRTP solving is not fast enough, indeed someone has to book his seat some hours before. So, in order to face the scalability problem, we propose a solution alternative to exact methods requiring too big computation times for a few requests. This stochastic solution is based on a genetic algorithm, which allows us to consider real-time CDRTP solving. This paper relates to a part of the research. We endeavour to show the methodological and operational relevance of a robust and flexible Convergent DRT (CDRT). Moreover, by using a genetic algorithm, we expect to provide a solution for solving a multi-objective scalable real-time CDRTP. Linked to the problem size (scalability), the real-time approach has to be considered with small problems (around 30,40 requests), and in the cases of larger problems (around 100 requests), a dynamic approach with the same multi-criteria optimization replaces the previous one.
In a first section, we present an outline of major related works to CDRTP, by focusing on exact or approached solving methods, describing the criteria and objectives to be taken into account, and enhancing their contributions and limits. Thereafter we explain the methodology used and exhibit the data-processing objects useful for the P&D N-TSP solving. We define and build the graph of convergence (CG), the associated total order, detail certain important properties and Sprawling Spanning Trees (SST). In the following section, we propose a mutation oriented genetic solution for solving the multi-objective CDRTP. Then we analyze the experimental projects, conclude and give a few prospective related works to be developed by

the TADvance group in order to reduce the complexity of coming CDRT supply.

2. The Convergent DRT problem

The Convergence Demand Responsive Transport (CDRT) consists in picking up a group of users, who have requested a trip and in carrying them to several specific points on the territory at a given hour. For example, the conference auditors, once they have arrived in the conference place could need to be served with a CDRT. Indeed, they all booked hotels more or less close to the conference place, and they have to arrive few minutes before the beginning of the conference. So, considering they first have expressed their wish, it is possible to calculate the path(s) of one (or several) vehicle(s), that is convergent to the conference place and responds to the whole set of requests by grouping the passengers. The optimization consists in finding the minimal number of vehicles used and travelled kilometers, and in the occupation rate of the vehicles.

On a more general point of view, the CDRT is an interesting way to serve attractive places of interest. It is often used during the edge times hours (very early or very late in the day) to complete the regular transport supply or on areas with a lack of services. The principal constraint is that users have to book their seat. This allows to process an asynchronous optimization (cars, drivers and paths) every day.

To be the less expensive possible and to respond to the users demand, a CDRT system has to work with fit-to-use and rigorously built data. The proposed paths have to be defined in terms of cost, and also service. For example, clients must not have the impression that they do not converge to the destination point. In our study, the available data are: the road network and its use conditions, the set of transport requests for a given time, the maximal duration of the route, the maximal delay at the pick-up point, and the margin of time necessary before the event time (i.e. we arrive 5 minutes before the conference begins).

Such a problem has indeed multiple aspects. We do not see here neither the previous stage for designing the service according to user needs, nor the final step concerning users management (information delivery and booking) and service operating (effective implementation of the paths and associated vehicles). We shall focus on two methodological aspects. First, the data used by the optimization algorithm enable to build a point-to-point matrix of times and distances from a road network, from which we identify and build a Convergence Graph (CG) to the convergent point. This CG is directed, acyclic and transitive. Secondly, we shall exhibit the underlying logical structure of the generic CDRT problem : the Sprawling Spanning Trees (SST). From a one-pass run on the graph, we can demonstrate that all valid solutions are computable and take the form of SST (cf. [14]).

3. Related work

The CDRT problem is a parent to the np-complete TSP, for which various solutions exist in accordance to the use cases. In its simplest configuration the CDRT problem is also linked to the Vehicle Routing Problem with Time Windows (VRPTW) [15]. In these two problems, the vehicles capacity can be taken into account, but also time windows in specific pick-up points on the territory. However, it is necessary to consider the delivery points with associated constraints. Finally, in a more complex DRT configuration, the problem can be associated to the Dial A Ride Problem (DARP). We shall be then interested in a generalization of the TSP, which corresponds more to the actual DRT set up, carrying clients to the destination point linked to the regular transport network. A generalization of this problem is the N-TSP, that we solved with N vehicles by partitioning, or spanning a territory into N smaller territories. The exact methods are not used because of their unsufficient robustness. Indeed the major drawback remains the scalability. Thus approximated methods are preferred, like approximation schemes proposed by Arora [18].

Another variant of the generalized TSP is the Pickup & Delivery TSP ([17], [3]), where goods and passengers are picked up and led to precise destinations. For that problem various forms also exist, to which different approaches are proposed [13] solving the generalization of this problem: the k-PDP. Other TSP forms combine the N-TSP to the k-PDP, for example the k-PD N-TSP [10], which generalizes the TSP and takes into account the capacity k of one vehicle. Unfortunately these approaches do not take into account neither the number N of vehicles, nor any QoS. Indeed, a more general solution of the DRT problem requires to consider the human and material resources [12]. Indeed, software solution exist, that must better respond to constraints imposed by the users requests, and include the available resources. Some proposals consider hazards (is a vehicle broken down?) and adapt turns in consequence.

The genetic approaches are preferred to Branch & Bound approaches when we address the problem of the scalability or when the knowledge of a problem and its resolutions are unsatisfactory. With these approaches, multi-objective evaluation functions could be apprehended. They apply strategies for solving np-difficult problems [8], especially those in relation to the transportation. Some works propose a real-time approach for optimizing vehicles networks ([5],[9]). Crossover and mutation operators reduce the number of used vehicles, without penalizing clients or increasing travel times.

About spanning trees, various approaches exist. We can cite the works of Raidl and Julstrom ([1],[7],[2]), who use evolutionary algorithms to encode spanning trees in undirected graph problems.

Our contributions depend on previous works [16], that describe a CDRT system using the ∞-PD TSP with n pick-up points $\leq m$ deliveries $= 1$ and solving simultaneously the N-TSP. The actual operational systems suffer from a lack of robustness (scalability) and flexibility. Indeed, they fail in proposing a simultaneous convergence to several points of a territory, by solving the ∞-PD TSP with $n \geq m \geq 1$ for a single vehicle and always by solving simultaneously the N-TSP.

4. Data representation

With a Geographical Information System, we build a point-to-point accessibility graph. By fixing the convergent point, we determine the vector of the departures scheduling for each graph vertex, and then the directed acyclic graph, that is the convergence graph (CG). We show then some properties of the CG.

4.1. Convergence graph

Let be a section of a road network, we determine the theorical time necessary to travel it. Theorical Time per Section (TTS) equals to the length of this section multiplied by the maximal speed authorized on the section: $TTS = length_{section} \times speed_{max}$. We determine next the "roughness" (the difficulty to move) of each section. In a first time, the physical roughness (Rp) due to physical parameters of the section, like the winding, the gradient, the quality of surface. Then we compute the roughness of usage (Ru) due to intensity of section usage in function of the hours of usage. Hence, we compute the theorical time of travel (TTT) from one point to another in the graph. We note $Mttp$ the matrix of theorical times of travel: $Mttp = Mttt + Mrp + Mru$, where $Mttt$: the amount, along the smallest paths, of the theorical times per section ; Mrp: the amount, along the smallest paths, of the theorical times due to the physical roughness ; Mru: the amount, along the smallest paths, of the theorical times due to the roughness of usage.
The data used to determine $Mttp$, though their good quality, do not allow to build an exact representation of the reality. We add a fourth term to the amount building $Mttp$. This term, denoted M_Δ, allows to introduce a correction reflecting the observed reality. If the three terms $Mttt$, Mrp and Mru are positive and represent times, the last term M_Δ could be positive, null or negative in function of observed gaps: $Mttp = Mttt + Mrp + Mru + M_\Delta$, with $Mttp$ the incidence matrix of the point-to-point accessibility graph (PPAG). In our case, each point represents a pick-up point and can be reached by any other point. PPAG is a complete directed graph.

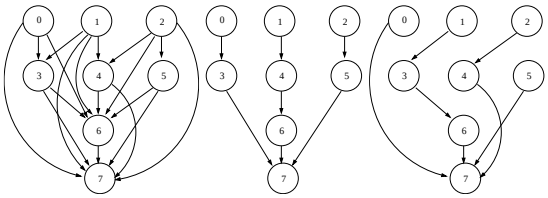

Fig. 1: Convergence graph and two SST

The CG is transitive. That is, if A and B are connected and, B and C are connected, so A and C are connected. That could be shown on the following manner: if A and B are connected ($HdB - HdA > Mttp(A, B)$) and B and C are connected ($HdC - HdB > Mttp(B, C)$), by summing these two expressions we have $HdC - HdB + HdB - HdA > Mttp(A, B) + Mttp(B, C)$. But $Mttp(A, B) + Mttp(B, C)$ is the upper bound of $Mttp(A, C)$. Indeed it is possible to pass by B to go to C from A, but it is also possible to find a smaller path. So, we find: $HdC - HdA > Mttp(A, B) + Mttp(B, C) \geq Mttp(A, C)$, that corresponds to the expression "A is connected to C". That proves the transitivity.
The CG is acyclic. An acyclic graph has no path passing twice a same vertex. So, we must prove, whatever A and B it exists a path from A to B, whereas it exists an edge from A to B. Let us suppose it exists a path from B to A and by transitivity it exists an edge from B to A, so: A and B are connected: $HdB - HdA > Mttp(A, B)$, and B and A are connected: $HdA - HdB > Mttp(B, A)$. By construction, we know all $Mttp$ values are positive. So $HdB - HdA > 0$ and the opposite value $HdA - HdB < 0$. That refutes that this value is greater than an element of $Mttp$, which is greater than 0. So, these two inequalities cannot be verified simultaneously. Hence the CG is acyclic.

4.2. Total order of the nodes

Besides the adjacencies list, the CG uses the total order of the nodes. For example, if we initialize a CG $G = (E, V)$ where $V = c, 2, 3, 4, 5$, c the convergence node, and $E = (4, c), (4, 2), (4, 3), (5, c), (5, 2), (5, 3), (2, c), (2, 3), (3, c)$ (E is the set of directed edges). The terninating paths set is empty, and the partial paths set either. The optimization consists in computing the total order of the vertices, on the one hand by defining the partial order of dependance and on the other hand, by refering to the decreasing order of the TTT to the convergence node. So, to compute the total order (O_t), we need to:
1. initialize $O_t = \{\}$ and $CG = G$;
2. determine the minimal elements of the CG. We sort then these elements in a decreasing TTT order and we add these sorted elements to $O_t(G) += O_t(CG)$;
3. compute $CG = CG - MinimalElements(CG)$;
4. iterate step 4.2 until the CG is empty.
Finally we get $O_t(G) = \{4, 5, 2, 3, c\}$.

5. Genetic algorithm

To solve CDRT problem, we propose a solution based on a genetic algorithm, that allows the evolution of a population of solutions to optimal solutions. Our algorithm is based on a steady state genetic algorithm, which creates a population by cloning the start genome. Each generation, it creates a temporary population, adds it to the already existing population, then removes the less interesting individuals (those whose fitness is the smallest) to get back to the initial size of population. This algorithm takes in input a graph described as an adjacencies list.

5.1. Objective definition

Let us define a standard solution and above all, what is a good solution in our study. First, we will see the representation we choose and how we initialize the beginning population. Then we will see the objective function of our problem.

Chosen representation We define a chromosome solution as an uni-dimensional array, where each cell (locus) represents a pick-up point and where the value of this cell (gene) indicates the number of the vehicle serving this point. So, the number of genes of the chromosome corresponds to the number of pick-up points (nb_{pkp}) minus the convergent point, i.e. $nb_{genes} = nb_{pkp} - 1$. Chromosome presented on table 1 gives a solution of the picking-up of the CG of figure 1. It corresponds to SST $((0, 3, 7); (1, 4, 6, 7); (2, 5, 7))$. Vehicles are indexed from 0 to $nb_v - 1$ where nb_v is the number of required vehicles to solve the CDRT problem.

pick-up point	0	1	2	3	4	5	6
vehicle number	0	1	2	0	1	2	1

Tab. 1: A chromosome solution of CG of figure 1

Initialization of the population The number of vehicles cannot be greater than the number of pick-up points and there are at least as many vehicles as initial pick-up points: $nb_{pkp} \geq nb_{vehicles} \geq nb_{pkp_{start}}$
We randomly initialize 1000 individuals by assigning a vehicle number among [0, nb_{pkp}]. Thus the start population has a number of not viable individuals (i.e. wrong solutions). Then the crossover and the objective contribute to improve the solutions.
N.B. for each chosen solution, there will be necessarily a different vehicle for each initial pick-up point (i.e. the minimal nodes). So, it could appear more interesting to arrange the table so that minimal nodes are located "on the left" of the table, according to the total order of the nodes. That is, each minimal node from 0 to k is respectively served by a vehicle number from 0 to k: $\forall pkp_{minimal_i}, n_{vehicle_i} = i, i \in \mathbb{N}$

Objective If nodes are correctly arranged, a good solution consists in checking the chromosome, that responds to the condition:

$$\forall i, j, i < j, gene(i) = gene(j) \quad (1)$$
$$\exists (i, j) \in E, H_{pkp_i} + t_{i \to j} \leq H_{pkp_j} \quad (2)$$
$$(1) \Rightarrow (2) \quad (3)$$

The fitness of a chromosome increases when it satisfies this condition. In an underlying manner, this favours solutions, that require the lesser number of vehicles, that is, a good solution minimizes the number of PSST. In the contrary, when a chromosome do not satisfy the objective condition, its fitness is reduced relatively to the error rate (another solution consists to give fitness 0).

Crossover operator Given that we have a different vehicle for each initial pick-up point, all chromosomes start with the same common subsequence S. So, to cross two individuals, it is not necessary to span this subsequence S. The crossover must happen at least from the first gene located over S. Let g_{l_S} be the first gene located over the common subsequence S.
Our crossover operator is parent of the PMX operator proposed by Goldberg [4]. This operator generates two children from two parents. The crossover is simply [6] completed by choosing two random cross points and by swaping sequences of the parents. For example, in the case of chromosomes P1 end P2 (cf. table 2), we choose a first random number k, $g_{l_S} \leq k \leq n$, corresponding to the start of the sequence to swap. We choose a second random number l, $k < l \leq n$. Then we create two children C1 and C2 from parents P1 and P2 by swaping the sequences $(kl)_{P1}$ and $(kl)_{P2}$ (cf. table 3).

0	1	…	i	j	**k**	**l**	m	n
0	1	…	i	x	***z***	***w***	y	v
0	1	…	i	j	**k**	**l**	m	n
0	1	…	i	w	***v***	***y***	z	x

Tab. 2: Parents chromosomes P1 and P2

6. Mutation operator

Mutation aims at bringing diversity into the current population. It is a random process realized from gene g_{l_S}. So, we assign a random value to a randomly chosen gene,

0	1	…	i	j	**k**	**l**	m	n
0	1	…	i	x	***v***	***y***	y	v
0	1	…	i	j	**k**	**l**	m	n
0	1	…	i	w	***z***	***w***	z	x

Tab. 3: Children chromosomes C1 and C2

modulo the number of vehicles in chromosome C, so that we do not increase the number of required vehicles:

$$\forall gene(i) \geq g_{l_S}, gene(i) \leq gene(|C| - 1), \quad (4)$$
$$gene(i) \leftarrow (gene(i) + val_{rnd}) \% nb_{v_{ch}} \quad (5)$$

Let us apply the mutation to chromosome I (table 4) ; if we randomly choose gene k, and we add the random value $v_{rnd} = 2$ to $gene(k)$, $gene(k) \leftarrow gene(k) + v_{rnd}$, we get $gene(k) = z$. After having applied *modulo* z, $z = nb_{v_I}$, we get $(gene(k) \leftarrow gene(k) \% z) = 0$. Although the SSGA tends the population to converge quickly to one pool of optimal solutions, the mutation operator increases the convergence more significantly.
So, with such a mutation mechanism, we always minimize or stabilize a criteria defined in the objective. Hence if we consider different objectives, a good solution will be a compromise of the best solutions, maximizing or minimizing the criteria at best.

0	1	…	i	j	**k**	l	m	n
0	1	…	i	x	**x**	w	y	v
0	1	…	i	j	**k**	l	m	n
0	1	…	i	x	***0***	w	y	v

Tab. 4: Example of mutation of gene k of individual I

7. Realizations and experiments

We proceed to the computation of the SST on various CG with increasing sizes, by using the SSGA described above. We have programmed this SSGA in C++ on a PC (2.4GHz with 4GB of memory) running Linux Debian with kernel 2.6. Table 5 presents the obtained computation times and reveals the interest of the exhaustive method for CDRT problems of size smaller than 17 pick-up points. Let us note some elements of the $k-PD\ N-TSP$ complexity. When the QoS constraints and the total order of the graph nodes are not taken into account, the complexity corresponds to generating all combinations of letters in a set of n letters ($O(2^{n-1} \times n!)$). By bounding the paths computation, we can limit complexity between $O(2^n)$ and $O(2^{an})$, with $a > 1$ and $n \geq 5$. For example, if we have a CG G with 7 nodes $|G| = 7$, the complexity equals to 203. Thus, for a great number of nodes, we need to use another kind of algorithms, such as meta-heuristic algorithms and more especially genetic algorithms. From a size greater than 40 nodes, it is laborious to realize a hand-made test, so we generate random convergent CG from a 50 nodes size to 1000 nodes size.
The application is programmed with the *Genetic Algorithm library* (GAlib) developped by the MIT [11]. We use a constant crossover rate $p_{cross} = 1.0$. Each result for one mutation rate is an average data realized on a 100 processes serial. This algorithm provides a set of representative solutions of good quality, in an acceptable computation time. The convergence speed, with the pertinency of the operators and an appropriate parameterizing, allows

to consider real-time applications for a size of problems closed to several dozens of nodes.

Number of nodes	Number of edges	CPU time
8	18	4.56
12	41	5.56
13	44	5.82
14	54	6.21
16	73	6.64
23	176	10.74
25	247	11.79
30	391	15.35
40	743	27.65
50	1156	65.88
60	1693	90.71
70	2305	116.94
80	3011	161.06
90	3821	215.83
100	4825	268.98
200	19569	1602.02

Tab. 5: Computation times (seconds) of SST

7.1. Study of variation of mutation rate

We vary the mutation rate p_{mut} from 0.0001 to 0.9 on a population of 1000 individuals, by solving the CDRT problem on CG with 20, 30, 40 vertices. In table 6, $|S|$ is the average number of different solutions and $\overline{D_{bs}}$ indicates the average percentage of optimal solutions obtained on these graphs. Globally, by increasing the mutation rate, we increase the number of various solutions. From a mutation rate $p_{mut} \simeq 0.1$, there is no more significative increase of the number of distinct solutions. Moreover, the more important the graph density is, the stronger the mutation rate must be if we expect a solution as soon as possible. Indeed, it is only from a mutation rate around 0.1, that the whole part of solutions are optimal, whereas in the cases of smaller graphs (20, 30 nodes) the population has great sets of optimal solutions with weaker mutation rates ($p_{mut} \leq 0.05$).
We converge quickly to SST minimizing the vehicles number, the travelled distances and the economic cost. The crossover and mutation operators, and also fitness lead us to affirm that, from a generation having perhaps only one SST, this one is sufficient if we hope to generate all solutions with the same number of vehicles or reducing it. A mutation rate equals to 1, that favours the representation of the best solutions in the population lead to the convergence, speeded up 7 times on examples with 40 nodes. That is why 6 generations are sufficent for the steady state genetic algorithm compared to another version of incremental genetic algorithm. A good parameterizing leading to the best solutions is still to be identified.
N.B. If we parameterize the algorithm with a null crossover rate $p_{cross} = 0$ and we use the mutation only (a strong rate $p_{mut} = 1.0$), the population converges quickly toward the optimal solutions pool. Here, the mutation is sufficient to evolve the population.

p_{mut}	CG_{20}		CG_{30}		CG_{40}	
	$\|S\|$	$\overline{D_{bs}}$	$\|S\|$	$\overline{D_{bs}}$	$\|S\|$	$\overline{D_{bs}}$
0.0001	29.0	90	103.3	80	8.9	0
0.0005	43.9	80	144.5	30	72.4	0
0.001	26.0	70	194.5	30	34.3	0
0.005	101.9	100	394.0	70	355.7	0
0.01	165.1	100	696.4	70	586.6	0
0.05	619.8	100	971.4	100	832.7	30
0.1	767.0	90	980.7	90	793.3	80
0.2	852.0	100	981.1	100	848.0	90
0.3	863.4	100	977.1	100	857.5	100
0.4	882.2	100	981.0	100	869.1	100
0.5	887.2	100	981.7	100	873.4	100
0.6	896.0	100	983.1	100	896.1	100
0.7	907.2	100	981.7	100	905.1	100
0.8	918.1	100	984.9	100	918.9	100
0.9	925.1	100	987.4	100	935.5	100

Tab. 6: Variation of number of different solutions

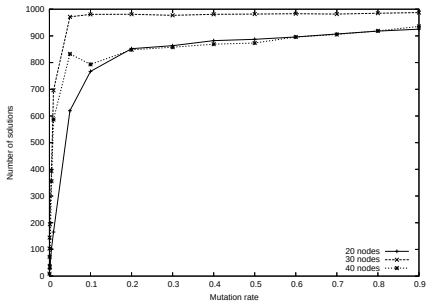

Fig. 2: Number of different solutions in varying mutation rate on CG with $\{20,30,40\}$ nodes

7.2. Study of convergence speed

We are now interested in the convergence speed and we want to know from which generation the first optimal solution appears. Generally, the more the mutation rate increases, the faster we have an optimal solution (cf. figure 3). The first optimal solution appearing is to correlate with the number of nodes and the edges density. Thus, for small dimensions problems like graphs with 20, 25 nodes we get an optimal solution very quickly (before around 30 generations in average) for a mutation rate $p_{mut} < 0.05$. But, as soon as we increase a little bit the nodes and edges density (30 nodes), the first optimal solution appears very later for a similar mutation rate (around 600 generations). In the case of the 30 nodes graph, it requires a mutation rate $p_{mut} \geq 0.05$ to find an optimal solution before 200 generations! The algorithm may find an optimal solution to the 30 nodes graph problem in around 30 generations with a stronger mutation rate only ($p_{mut} \geq 0.3$).

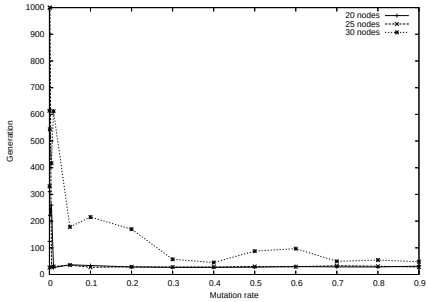

Fig. 3: Average generations of appearing of the first optimal solution

7.3. Convergence to local pools

As DeJong explained in his first works (cf. [4]),using a steady state algorithm tends to gather the solutions to a localized pool among the optimal solutions pools, according to a random factor. By defining more heuristics, we reduce the set of optimal solutions, but also the number of optimal solutions pools. Although the following CG is

unrealistic of a CDRT problem, we consider it with and without constraints (travel times or distances) on edges to illustrate this particular point:

without constraints	with constraints
0: 2 3 4 5 6 7 8 9	0: 2/2 3/3 4/4 5/3 6/5 7/6 8/7 9/8
1: 3 4 5 6 7 8 9	1: 3/2 4/1 5/3 6/3 7/7 8/7 9/8
2: 5 7 8 9	2: 5/1 7/4 8/4 9/6
3: 5 6 7 8 9	3: 5/1 6/2 7/5 8/4 9/5
4: 6 8 9	4: 6/2 8/3 9/6
5: 7 8 9	5: 7/4 8/3 9/4
6: 8 9	6: 8/1 9/4
7: 9	7: 9/2
8: 9	8: 9/2
9	9

If we do not consider edges values, we have three pools of solutions, as follows:

1. ((0,3,5,7,9);(1,4,6,8,9);(2,9)), ((0,3,5,8,9);(1,4,6,9);(2,7,9)), ((0,3,5,9);-(1,4,6,8,9);(2,7,9)), ((0,3,6,9);(1,4,8,9);(2,5,7,9)), ((0,3,7,9);(1,4,6,8,9);(2,5,9)), ((0,3,7,9);(1,4,6,9);(2,5,8,9)), ((0,3,8,9);(1,4,6,9);(2,5,7,9)), ((0,3,9);(1,4,6,8,9);-(2,5,7,9))

2. ((0,2,5,7,9);(1,4,6,8,9);(3,9)), ((0,2,5,7,9);(1,4,6,9);(3,8,9)), ((0,2,5,7,9);-(1,4,8,9);(3,6,9)), ((0,2,5,7,9);(1,4,9);(3,6,8,9)), ((0,2,5,8,9);(1,4,6,9);(3,7,9)), ((0,2,5,9);(1,4,6,8,9);(3,7,9)), ((0,2,7,9);(1,4,6,8,9);(3,5,9)), ((0,2,7,9);(1,4,6,9);-(3,5,8,9)), ((0,2,8,9);(1,4,6,9);(3,5,7,9)), ((0,2,9);(1,4,6,8,9);(3,5,7,9))

3. ((0,2,5,7,9);(1,3,6,8,9);(4,9)), ((0,2,5,7,9);(1,3,6,9);(4,8,9)), ((0,2,5,7,9);-(1,3,8,9);(4,6,9)), ((0,2,5,7,9);(1,3,9);(4,6,8,9)), ((0,2,5,8,9);(1,3,7,9);(4,6,9)), ((0,2,5,9);(1,3,7,9);(4,6,8,9)), ((0,2,7,9);(1,3,5,8,9);(4,6,9)), ((0,2,7,9);(1,3,5,9);-(4,6,8,9)), ((0,2,9);(1,3,5,7,9);(4,6,8,9))

For this CG, the thousand of solutions will be grouped on one of these set of solutions, which could appear during an evolution. However, if we add a new heuristic consisting in minimizing the amount of the edges values of one SST, we have only one pool of optimal solutions (smaller than previous pools in term of solutions count):

((0,2,5,7,9);(1,4,6,8,9);(3,9)), ((0,2,7,9);(1,4,6,8,9);(3,5,9)), ((0,2,9);(1,4,6,8,9);-(3,5,7,9))

8. Conclusion and future work

This paper presents a mutation oriented genetic algorithm designed for solving the CDRTP. This algorithm aims at providing pragmatic solutions and flexibility to the system. Laying on methodically built objects (transitive graph, SST), the genetic algorithm takes the QoS constraints into account, and foresees also the implementation of the different criteria in a multi-objective approach.

The convergent mutation operator allows to balance solutions for serving at best the requests, according to different heuristics to take into account.

The tested experiments and results lead us to pursue our work by integrating new heuristics like capacity or time windows, and hence by continuing to develop a multi-objective approach. We want also to give the system more flexibility and to improve the algorithm to process greater floods, but always by willing to keep the real-time solving of the problem.

REFERENCES

[1] B. Julstrom and G. Raidl. Initialization is robust in evolutionary algorithms that encode spanning trees as sets of edges. In *Symposium on Applied Computing*, pages 547–552. ACM Press, 2002.

[2] Bryant A. Julstrom and Günther R. Raidl. A permutation-coded evolutionary algorithm for bounded-diameter minimum spanning tree problem. In *2003 Genetic and Evolutionary Computation Conference's Workshops Proceedings, Workshop on Analysis And Design of Representations*, 2003.

[3] P. Chalasani and R. Motwani. Approximating capacitated routing and delivery problems. Technical report, Stanford University, 1995.

[4] D. Goldberg. *Genetic algorithms in search, optimization, and machine learning*. Addison-Wesley, 1989.

[5] D. Montana, M. Brinn, G. Bidwell, and S. Moore. Genetic algorithms for complex, real-time scheduling. In *IEEE Conference on Systems, Man and Cybernetics*, volume 3, pages 2213–2218, 1998.

[6] D. Whitley. A genetic algorithm tutorial. *Statistics and Computing*, 4:65–85, 1994.

[7] Günther R. Raidl and Bryant A. Julstrom. Greedy heuristics and an evolutionary algorithm for the bounded-diameter minimum spanning tree problem. In *SAC '03: Proceedings of the 2003 ACM symposium on Applied computing*, pages 747–752, New York, NY, USA, 2003. ACM Press.

[8] K.A. DeJong and W.M. Spears. Using genetic algorithms to solve np-complete problems. In *Proceedings of the Third International Conference on Genetic Algorithms*, pages 124–132, 1989.

[9] M. Bielli, M. Caramia, and P. Carotenuto. Genetic algorithms in bus network optimization. *Transportation Research*, C:19–34, 2002.

[10] M. Forbes. Capacitated vehicle routing and the k-delivery n-traveling salesman problem. Master's thesis, Montgomery BlairHigh School, 2004.

[11] M. Wall. Genetic algorithm library. http://lancet.mit.edu/ga/.

[12] M.E.T. Horn. Fleet scheduling and dispatching for demand-responsive passenger services. *Transportation Research*, 10(1):35–63, 2002.

[13] M.W.P. Savelsberg and M. Sol. The general pickup and delivery problem. *Transportation Science*, 1(29):17–29, 1995.

[14] N. Christophides, A. Mingozzi, and P. Toth. Exact algorithms for solving the vehicle routing problem based on spanning trees and shortest path relaxations. *Mathematical Programming*, 20:255–282, 1981.

[15] Olli Bräysy and Michel Gendreau. Vehicle routing problem with time windows, part I and II. *Transportation Science*, 39(1):104–118 and 119–139, 2005.

[16] P. Canalda, P. Chatonnay, and D. Josselin. Énumération d'arbres couvrants tentaculaires, une solution au problème de transport à la demande en convergence. In *IEEE International Conference on Sciences of Electronic, Technologies of Information and Telecommunication, SETIT*, pages 146–154, 2004.

[17] J. Renaud, F. F. Boctor, and J. Ouenniche. A heuristic for the pickup and delivery traveling salesman problem. *Comput. Oper. Res.*, 27(9):905–916, 2000.

[18] S. Arora. Polynomial time approximation schemes for Euclidean traveling salesman and other geometric problems. *Journal of the ACM*, 45(5):753–782, 1998.